REMARKS

Claims 1-19 are currently pending in the patent application. The Examiner has rejected Claims 1-19 under 35 USC 103(a) as being unpatentable over Smith in view of Cowan. For the reasons set forth below, Applicants believe that the invention is patentable over the cited art.

The present invention is directed to a memory device, a memory pattern, system and method for preventing stack smashing attaches while a computer is executing a program. The invention provides for storage of a guard variable between a previous frame pointer and a data array stored in a local variable storage area. The guard variable is used as a target to confirm whether a return address, pointed to by the frame pointer, has been destroyed. It is integral to the invention that the guard variable be stored in the stack between the previous frame pointer and the data stored at the local variable storage area, or else the guard variable itself can be compromised during a stack smashing attack. With this arrangement, not only the return address but also the previous frame pointer is protected and system intrusions and smashing attacks can be completely prevented.

The Examiner has cited the combined teachings of the Smith and Cowan articles against the presently claimed invention. The Smith article is cited for its specific storage configuration including a return address storage area, a previous frame pointer storage area, and a local variable storage area located below the return address storage area and the previous frame pointer storage area. As acknowledged by the Examiner, the Smith article fails to disclose the use of a guard variables. In essence, therefore, the Smith article simply illustrates stack vulnerabilities, without providing any suggestion that the storage configuration be modified to address the stack vulnerabilities. In fact, Smith effectively teaches away from such a suggestion by teaching that "...changes must be implemented in the privileged programs themselves, in the C programming language compilers, or in the operating system kernel" (page 17, section 11. STACK SMASHING PROTECTION, first paragraph). Applicants respectfully assert that the Smith article does not suggest the invention, does not suggest modification of its disclosed storage configuration, and does teach away from such modification.

In acknowledging that the Smith article does not disclose the use of a guard variable, the Examiner has cited the Cowan article. Cowan details the StackGuard approach, which was discussed in detail in the background section of the present Specification. StackGuard provides for a so-called "canary word" to be placed next to the return address on the stack. According to the StackGuard program described in the above mentioned reference, the guard_value is stored "next to" the return address. As illustrated in Cowan's Fig. 2, the canary work is stored between the return address to be used to return to a preceding function, which is the calling source for the function that is currently being executed, and the local variables storage area. Thus, while the return address is protected, a frame pointer that is used to describe the scope of the variable is not, and this constitutes a serious security hole. Since the frame pointer is not protected, intrusive control of the program counter is possible.

As detailed in the present Specification, in the function return processing, the following transactions take place.

FP $\rightarrow$ SP

PFP $\rightarrow$ FP

(SP) $\rightarrow$ PF; return operation

where, FP denotes a frame pointer, SP denotes a stack pointer and PC denotes a program counter; an arrow is used to indicate the substitution of a value; and the parentheses are used to identify a direct reference.

As is shown above, since the StackGuard program does not protect the PFP in a stack, the value of the PFP can be controlled by an external attack. And when the transmission of the value is examined and the above function return processing is referred to, only the PFP need be used to control the frame pointer at the first return and to control the program counter at the second return. While actually a test based on the guard_value must be passed in order to embed malicious code, a stack smashing attack can be delivered to the guard_value that is used as a default by the StackGuard. Furthermore, a method for attacking the other guard_value may also be found. Therefore, the conventional method for which the StackGuard program is used can not completely prevent a stack smashing attack.

In addition, according to the StackGuard program, the validity of the guard value must always be confirmed at the time of the second return in order to detect to what degree the frame pointer is affected that was damaged at the first return. Therefore, for all the functions the program is produced to confirm the validity of the guard value, or canary word. As a result, overhead is always incurred because of the process for confirming the validity of the guard value. Further, since according to the StackGuard program the guard value is inserted between the return address area and the PFP area, the frame structure of the stack is changed, and therefore, debugging support can not be provided for a program for which the StackGuard is mounted.

The Examiner has concluded that it would be obvious to modify the Smith storage configuration, wherein a previous frame pointer is found after the return address, with the canary word of Cowan. Applicants respectfully disagree. Firstly, as discussed above, Smith teaches away from modification of the storage configuration for guarding against stack smashing attacks, and teaches modification of programming, etc. as the appropriate mode for preventing stack smashing attacks. Clearly, therefore, neither suggestion nor motivation is provided for modifying the Smith storage configuration. Moreover, Cowan expressly teaches that the canary work be placed "next to" the return address. As detailed above, such an arrangement is unworkable for guarding against certain stack smashing attacks. Moreover, even if one were motivated to place the Cowan canary word next to the return address in the Smith storage configuration, one would not arrive at the present invention as claimed. The present claims expressly recite that the guard variable be placed between the previous frame pointer and the data array in the local variable storage area, so that both the previous frame pointer and the return address can be protected. Applicants contend that the combination of references does not render the claimed invention obvious and respectfully request reconsideration of the rejection based on the combined teachings of Smith and Cowan.

In light of the foregoing amendments and remarks, Applicants respectfully request entry of the amendments, reconsideration and withdrawal of the rejections, and issuance of the claims.

Respectfully submitted,

H. Etoh, et al

By: Anne Vachon Dougherty

Anne Vachon Dougherty
Registration No. 30,374
Tel. (914) 962-5910